

Decentralized Random Walk-Based Data Collection in Networks

Iqra Altaf Gillani¹, Amitabha Bagchi¹, Pooja Vyavahare², and Tanmay Gupta³

¹{iqaaltaf,bagchi}@cse.iitd.ac.in, Department of Computer Science and Engineering, IIT Delhi

²pooja_vyavahare@iitmandi.ac.in, School of Computing and Electrical Engineering, IIT Mandi

³tanmay.gupta100@gmail.com, Rivigo Inc.

February 2, 2017

Abstract

We analyze a decentralized random-walk based algorithm for data collection at the sink in a multi-hop sensor network. Our algorithm, Metropolis-Collect, which involves data packets being passed to random neighbors in the network according to a simple metropolis random-walk mechanism, requires no configuration and incurs no routing overhead. To analyze this method, we model the data generation process as independent Bernoulli arrivals at all nodes except the sink. We analyze both latency and throughput in this setting, providing a theoretical upper bound for latency and a theoretical lower bound for throughput. While our random walk-based method is not comparable to routing-based methods in terms of latency, we prove that it is theoretically no more than a logarithmic factor higher than the worst case latency. The main contribution of our paper, however, is the throughput result: we show that the rate at which our algorithm can collect data is lower bounded by the spectral gap of the random walk's transition matrix, which is, in turn, proportional to the spectral gap of the network for regular graphs. Through simulations we compare our rate bounds to the routing-based directed diffusion method, showing that although our method achieves a lower throughput (while incurring no setup cost), the difference in the two methods is directly proportional to the degree of the network.

Keywords: Data collection, throughput, random walk, sensor networks

1 Introduction

Sensor networks are useful in applications, for the environment and habitat monitoring, disaster recovery, agricultural monitoring, health administration, and target-tracking [3], where human intervention is not possible. Being unattended these networks are more prone to frequent topology changes due to a random node or link failures. Such scenarios demand low overhead, more robustness and fault tolerant algorithms. Thus topology-based algorithms are not useful for these networks as they involve high overhead for maintenance of topology information and recovery mechanisms for critical points of failure [30]. On the other hand, stateless algorithms like the random walk-based algorithm we present in this paper, are reliant only on local information, have low overhead and do not suffer from the problem of critical point failure (e.g. cluster heads [17] or nodes close to root of data gathering tree [12]) as all nodes of the network play similar role in the algorithm (see e.g. [5]).

These algorithms are simple, fault tolerant and scalable, thus more suitable for unattended sensor networks. Despite all these advantages random walk-based algorithms are not often preferred for data collection since they tend to have higher latency compared to routing-based methods which rely on global topology information and are vulnerable to changes in topology but are optimized for low latency.

However, some sensor network applications particularly those involving monitoring of habitat, environment or agriculture like Intel’s Wireless Vineyard, Great Duck Island project, Zebrant project, and many others (see [35] and references therein) can compromise on latency but require uninterrupted continuous data collection. Motivated by these examples, we propose a simple decentralized random walk based algorithm Metropolis-Collect, for data collection which requires no configuration and thus has zero setup cost.

We consider a multi-hop sensor network where each node is equipped with a queue (we use the term buffer and queue interchangeably) which helps in store and forward process of data. Each node gathers data at a fixed rate from its surroundings and stores it in its buffer along with other data packets that may be generated earlier or received from neighboring nodes. At each time-step, each node picks a random data packet from its buffer and forwards it to a randomly chosen neighbor. Metropolis-Collect algorithm does not require any global information and is truly distributed. We study the movement of data packets in Metropolis-Collect by modeling it as a random walk on the graph representing the underlying network. We provide theoretical bounds on the latency and throughput for Metropolis-Collect. Though our latency bound is not comparable to that of routing based algorithms, we observe that it is no more than logarithmic factor higher than worst case latency of routing based algorithms. We study the throughput of the network and provide a lower bound in terms of the spectral gap of the transition matrix of a Metropolis random walk on the graph and an upper bound in terms of the edge expansion of the graph. We also demonstrate the tightness of our throughput bounds through simulations for random d -regular graphs and random geometric graphs. Through simulation we compare the Metropolis-Collect algorithm with directed diffusion [21] which is known for its high throughput and show that although directed diffusion achieves better throughput than our algorithm, its advantage over Metropolis-Collect is bounded and depends linearly on the degree of the network.

Our Approach Given a sensor network, we assume that each node except a designated node called *sink* is sensing the environment. Specifically, each node senses data as independent Bernoulli process with some *stable* rate and relay it to the *sink*. Stable rate ensures that all data is successfully collected at the sink (we will define it more formally in section 4.1). In our model, we assume that the network is connected, but we do not expect any node to know anything about the network except the identity of its neighbors (the nodes with which it can directly communicate). We also assume that time is slotted and nodes communicate with each other at the start of every time slot. However, this assumption can be easily removed and does not affect our results.

In Metropolis-Collect algorithm at the start of any time slot, a node picks up a data packet from its queue uniformly at random and forwards it to a neighbor who is also chosen uniformly at random. We allow a node to transmit only one packet to one of its neighbors, but, it can receive multiple packets from its neighbors. This is known as transmitter gossip constraint [8, 32] and has been used in literature [25, 8, 26, 32] for energy conservation and to prevent data implosion [18]. The movement of any data packet in such setting can be seen as the *random walk* of the data packet on the graph. However, presence of other packets in the buffer of a given node results in delaying of the given packet, thereby, making analysis of random walk harder. So, to analyze it we need to ensure that the expected queue size is bounded by 1 for all nodes at all time steps (discussed in

detail in section 3.3). For regular networks such constraint is easily satisfied but for ensuring it for general graphs we use a modified version of random walk known as Metropolis chain wherein each node is allowed to accept or reject a data packet from its neighbor with some probability. This modified version of random walk is also helpful in achieving load balancing [5] among the sensor nodes.

We define a parameter, *collection time*, which is the time taken for the sink to receive data packets from all the nodes and hence, represents the latency in collecting all data. We also define the *throughput of network* which is the rate at which data is received at the sink. We analyze these parameters by studying the movement of data packets as the random walk on the graph representing the underlying network. For a single round of data collection scenario, we study n random walks moving in the network (where n is the number of nodes in the network) and determine the collection time in terms of graph parameters like maximum, minimum degrees and worst-case hitting time. Similarly, for a stable data arrival rate, we find the average data collection time over k rounds of data packets sensed by each node.

As each node in Metropolis-Collect algorithm makes transmission decisions based on its immediate neighborhood (local information), it does not require any global information about the network topology or position of the sink node. The algorithm requires zero configuration before it starts data transmission and is immune to topology changes during its execution.

The major drawback of our work is that our transmission model allows simultaneous transmission and reception and also allows for a node to receive more than one packet at a time, thereby bypassing the question of interference which is critical to sensor networks built on wireless nodes. This effectively means that at first sight, it appears that our results are valid only for wired sensor networks. However, this is not so. As we discuss in Section 7, we feel that our mathematical techniques can be adapted to a setting where a node can either receive or send in one slot and where a node can receive from exactly one neighbor in a given slot. We feel that the bounds we prove will not change qualitatively in such a setting.

Our contributions

1. We propose a simple, low overhead, fault-tolerant decentralized random walk based algorithm, Metropolis-Collect, to collect data packets from all the nodes at the designated node *sink*.
2. We give an upper bound on the latency for Metropolis-Collect and show that the data rate which determines the network throughput is lower bounded by the spectral gap of Metropolis-Collect's transition matrix which is proportional to the spectral gap of the network for regular graphs.
3. We study through simulation how close our throughput bounds are to the observed data rate and also compare the optimal throughput of our method to the throughput obtained by the well-known routing-based data collection method directed diffusion.

Organization Following a survey of the literature in Section 2, we formalize our network setting and data collection model in Section 3. We also present the Metropolis-Collect in Section 3. We define our performance metrics and main results in Section 4. In Section 5 we prove our main theorems followed by a discussion about the results and some examples. Some empirical results are presented in Section 6 along with simulations for analyzing our rate bounds and comparison with directed diffusion method. We conclude the work in Section 7 with a discussion for possible future work.

2 Related Work

Data Collection Algorithms Data collection in a sensor network is a well-studied field, and several algorithms have been proposed for it in the literature. We briefly discuss some of them and compare with Metropolis-Collect algorithm.

One category of such algorithms is location-based like GPSR [24] which uses some routing information at every node generated either using global or geographical information of the network. In a typical sensor network, routing information at any node becomes invalid frequently (due to movement or failure of nodes), hence these algorithms are not very efficient in practice. On the other hand, in Metropolis-Collect every node needs only to know about its neighborhood, thus is more robust and less vulnerable to node failures and network changes.

Another class of algorithms is based on hierarchical structures like clusters in LEACH [17] or chain in PEGASIS [29]. These algorithms also require time to learn some global information of the network and set up the cluster heads or chain of sensor nodes. Also, in clustering protocols [1], over a period of time cluster heads become the bottleneck for the propagation of data packets in the network and such solutions not scalable. The Metropolis-Collect is decentralized in the sense that each node transmits data only to its neighbors and there is no centralized node to govern these transmissions. Thus, Metropolis-Collect is truly distributed and scalable.

Data correlation [12] and coding techniques [23] have also been used for data collection. We do not use any coding technique in our algorithm and so we need low configured nodes which are just capable of storing and forwarding data. However, our network model is similar to Kamra et al. [23] as they use only local information in their algorithms and have a single sink for data collection.

Closest to the Metropolis-Collect algorithm is the technique used in gossip algorithms for data dissemination and rumour spreading [25, 8, 32]. In these algorithms, at any time step, a node selects one of its neighbor uniformly at random and communicates some data to it. Every node performs certain computation on the received data (along with its data) and transmits the output to a randomly chosen neighbor in the next time slot. We use a similar approach to communication in our setting, but unlike gossip algorithms, our aim is to collect all the data packets at the sink node. In Metropolis-Collect algorithm the nodes do not perform any computation on the data and just forward the data packets which they receive. Most of the literature in gossip algorithm setting computes functions like the average, sum or separable functions [32]. We are interested in collecting all the data packets which can be seen as *identity function* on the data. Moreover, we use push mechanism for spreading information rather than other variants like pull, push-pull as done by Demers et al. [13] and Karp et al. [25].

Random Walk Based Algorithms Models based on simple Random walks have been used in literature for query processing [5], to model routing for data gathering [30] and for opportunistic forwarding [10], but no analysis has been done for finding the average data collection time or resulting throughput explicitly. In a different context than ours, Neely [33] showed that when independent Markov processes modulated arrival processes, the average network delay grows at most logarithmically in the number of nodes in the network. Our latency (or data collection time) bounds are similar to these results.

Biased random walks have also been explored to improvise various results. One such biased random walk wherein priority is given to unvisited neighbors has been used by Avin et al. [5] to improve their query processing results and partial cover time bounds. They suggest future improvement *super bias* [19] which is similar to the Metropolis algorithm that we have used. Regarding the inherent problem of latency in random walks, they suggest performing parallel random walks as part of their future work[4]. In Metropolis-Collect, we consider data packets are performing

parallel random walks which reduce the latency considerably.

Network Throughput and Data Rate Data arrival rate determines the frequency at which nodes in the network are collecting data from their surroundings, hence the network throughput. It is a critical performance parameter of the sensor network. The throughput capacity of the network in different contexts has been thoroughly studied [14, 15, 7, 31] following the pioneering work by Gupta and Kumar [16]. Many protocols have been proposed which offer high data rates, most popular among them are Directed Diffusion [21] and its variants like Rumor routing [9], Energy-aware routing and others (see [2] and references therein). All these offer high data rates along a reinforced path but suffer from overhead in setting up and maintaining such path. Also, these are only suitable for query-driven models and not useful for applications which require continuous data gathering [2].

The stable data rate in terms of different underlying graph primitives has been found under varying contexts in literature. Banerjee et al. [6] determine the maximum data or refresh rate for a network computing *Fully-Multiplexible* functions in terms of min-mincut of the graph. Their result is a natural upper bound on our rate results as we simply collect data without any aggregation. Dependence of rate on the maximum degree of sensor network organized as a tree has been found by Incel et al. [20] in their setting. We also find a sufficient bound on the rate in terms of graph parameters like maximum and minimum degrees.

3 Model and Algorithm

In this section we discuss our network setting, and data collection model. We present the Metropolis-Collect algorithm in detail here.

3.1 Network setting

Sensor node capabilities We consider a connected multi-hop sensor network comprising a large number of sensor nodes deployed over a given geographical area. Each node of the network has a sensor which senses the environment, a transceiver which is used to send and receive data from other network nodes, and some storage in which data sensed from the environment and received from neighbouring nodes can be stored. The network is deployed with zero configuration i.e. no node is aware of the sink position or global network routes. Moreover, each sensor is provided with a standard pseudo-random number generator, which is used for generating random numbers to choose among the neighbors. The ability of sensor nodes to choose random numbers has been exploited a lot in literature, be it in Data-centric protocols like ACQUIRE, Gradient-based routing, and Energy-aware routing or hierarchical protocols like LEACH (see [2] and references therein).

Transceiver assumptions Each node is configured to forward data to one of its neighbor chosen uniformly at random, but it can receive data from multiple neighbors at any time step. This is known as transmitter gossip constraint [8, 32]. It has been widely used in literature [25, 8, 26, 32] as it results in slow energy dissipation and also prevents data implosion by maintaining only a single copy of data at any node [18, 2]. Multi-packet reception and simultaneous transmission-reception are easily ensured for wired networks due to the physical connections, but, for wireless networks, these are not possible due to interference. However, the main ideas underlying our proofs do not change even if we consider more realistic assumptions like allowing either transmission or reception of a single packet in a given time slot (see Section 7) and the analysis we present in this paper can serve as a benchmark for those scenarios.

Data generation and collection We assume that sensor nodes generate new data as required, e.g., a temperature sensor may be configured to generate a new reading when the change over the prior reading is at least a certain threshold value. This data is then relayed through the network to one of the nodes that is designated as the data sink and is responsible for collecting data that reaches it. Depending on the application, we assume the sink can then relay the data to some decision making or processing unit using *data mules* to monitor events, perform local computation or configure local and global actuators [9].

Network setup and routing We do not use any centralized algorithm to create connections among the nodes. At deployment time, sensors opportunistically make connections with every other sensor that they can directly communicate with. In wired networks, such nodes (neighbors) are the directly connected sensor nodes, and in wireless networks, these are nodes which lie within the transmission range of the given node. Our algorithm requires knowledge of the number of neighbours (the degree in graph-theoretic terms) of each of its neighbours so when the initial phase of making connections ends, each node exchanges this information with each of its neighbours. As nodes may become unavailable frequently, due to failures or enter into sleep mode (like in wireless sensor networks), nodes need to perform these handshakes periodically to ensure updated degree information of their neighbors (discussed in detail in Section 3.2). Hence, our scheme for network creation offers a basic communication mechanism without incurring any computational overhead. Moreover, no localization algorithm is required for establishing the multi-hop communications.

3.2 Network and Data Collection Model

Network model We model the multi-hop sensor network by an undirected graph $G = (V, E)$, where V is the set of n sensor nodes with one sink, u_s , and E is the set of edges. There is an edge $e = (u_i, u_j)$ between nodes $u_i, u_j \in V$, if u_i, u_j can directly communicate with each other. The neighborhood of a node u_i is the set of all nodes $u_j \in V$ which can communicate with it, denoted by $u_i \sim u_j$, i.e., $\text{Nbd}(u_i) := \{u_j \in V | (u_i, u_j) \in E\}$. The degree of u_i is $\deg(u_i) := |\text{Nbd}(u_i)|$. We denote the maximum and minimum degree among all nodes in the network by d_{\max} and d_{\min} respectively.

Time model We consider a synchronous time model wherein time is slotted across all nodes in the network and nodes communicate with each other at the start of every time slot. Our results do not depend on synchronization and can be adapted to the asynchronous setting as well but in this paper we present the synchronous setting for ease of presentation.

Data generation model We model the data generation process at each node as a stochastic arrival process in discrete time that is Bernoulli with parameter β and independent of the arrivals taking place at all other nodes, i.e. at each time slot t each node generates a new data packet with probability β independent of all other nodes. The sink has no data arrival taking place.

Store and forward model At any time slot t , due to the enforced transmitter gossip constraint, each node can send only a single data packet to a chosen neighbor, but each node can receive multiple data packets simultaneously from its neighbors. We also allow a node to send and receive at the same time. We have discussed the implications of this assumption in Section 1 and will further discuss how to remove this assumption in Section 7.

At every time step, each node maintains a queue of packets, either generated at the node itself or received from neighbours, which have not been forwarded yet. We denote the number of data

packets in the queue of node u_i , also referred to as the queue size, at the start of slot t by $\mathcal{Q}_t(u_i)$ and let $\mu_t(u_i) = \mathbb{E}[\mathcal{Q}_t(u_i)]$ be the expected queue size. Note that this expectation is over the random arrivals and the random choices made by our algorithm which will be explained further in the subsequent section.

3.3 Metropolis-Collect Algorithm

In our proposed algorithm Metropolis-Collect, at any time slot t each node chooses a data packet from its queue and transmits it to a randomly chosen neighbor. The movement of a data packet in the network can be seen as a random walk on the graph G , however the delay caused by the presence of other packets in the buffer of a given node makes this random walk an inhomogeneous Markov Chain which is harder to analyze. As we will see below it is possible to analyze this inhomogeneous chain but to do so we need to fulfill a technical requirement: we need to ensure that the expected queue size $\mu_t(u_i)$ is bounded by 1 for all nodes at all time steps. This requirement is easily fulfilled by the simple random walk on a d -regular graph but for general graphs we need a slight modification that uses the notion of a Metropolis chain [28, Section 3.2].

As we see in the formal definition of algorithm Metropolis-Collect, this modification is as follows: having selected a packet to forward from its buffer, a node u_i picks a neighbour, say u_j , uniformly at random from its $\deg(u_i)$ neighbours, but it actually forwards the packet to u_j with probability $\min\{\deg(u_i)/\deg(u_j), 1\}$. Formally, the transition probability from node u_i to u_j in the Metropolis chain is given by

$$\mathcal{P}[u_i, u_j] = \begin{cases} \frac{1}{\deg(u_i)} \left[\frac{\deg(u_i)}{\deg(u_j)} \wedge 1 \right] & \text{if } u_i \neq u_j, \\ 1 - \sum_{u_k: u_k \neq u_i} \frac{1}{\deg(u_i)} \left[\frac{\deg(u_i)}{\deg(u_k)} \wedge 1 \right] & \text{if } u_i = u_j. \end{cases} \quad (1)$$

In effect what this means is that if u_j has lower degree than u_i , the packet is always forwarded but if it has higher degree then it may not be forwarded. Intuitively we see that this prevents the packets from concentrating at high degree nodes, and, in fact, this selection of parameters ensures that the Metropolis random walk has a uniform stationary distribution (as opposed to the simple random walk in which the stationary probability of being at a node is proportional to the degree of the node).

Algorithm Metropolis-Collect Algorithm run by node u_i

Require: Node $u_i \in V$, which has neighborhood $\text{Nbd}(u_i)$, degree $\deg(u_i)$, non-empty queue at time slot t i.e. $\mathcal{Q}_t(u_i) > 0$ and knows updated values of $\deg(u_j), \forall u_j \in \text{Nbd}(u_i)$ due to regular updates from u_j , handshakes done at the time of deployment and then at periodic intervals.

- 1: Node u_i picks a packet from its queue with probability $\frac{1}{\mathcal{Q}_t(u_i)}$
 - 2: Picks a neighbor u_j from $\text{Nbd}(u_i)$ with probability $\frac{1}{\deg(u_i)}$.
 - 3: Node u_i sends the data packet to node u_j , which accepts packet with probability $\left(\frac{\deg(u_i)}{\deg(u_j)} \wedge 1 \right)$.
 - 4: Node u_j sends ACK to node u_i if it accepts packet, otherwise sends NAK. Both ACK and NAK are piggybacked with updated degree of u_j .
 - 5: Node u_i on receiving ACK from u_j deletes data packet sent to node u_j from its queue and does nothing if it receives NAK.
-

At the end of the time slot, if node u_j accepts the data packet and sends an ACK then the data

packet gets stored in u_j 's queue and is deleted from u_i 's queue. Otherwise, u_j sends a NAK and none of the queues are changed.

Expected queue size equation Since the expected queue size plays a key role in our analysis, we present it separately here. For Metropolis-Collect with transition probability $\mathcal{P}[\cdot, \cdot]$, running on a network in which each node is generating data at rate β as discussed in Section 3.2, the expected queue size at node $u_i, \forall u_i \in V \setminus u_s$ at time slot $t + 1$, is

$$\mu_{t+1}(u_i) = \mu_t(u_i) - \sum_{u_j: u_j \sim u_i} \mu_t(u_i) \mathcal{P}[u_i, u_j] + \sum_{u_j: u_j \sim u_i} \mu_t(u_j) \mathcal{P}[u_j, u_i] + \beta \quad (2)$$

Information maintenance Sensor nodes obtain the degree information of their neighbors using handshakes done at the time of deployment and then at periodic intervals. These intervals are fixed based on the application at hand and deployment scenario. To increase the interval duration and avoid frequent handshakes each node can piggyback its degree information while sending ACK or NAK. Piggybacking ensures updated information in between the intervals while incurring no extra overhead.

4 Metrics and Analysis

In this section, we first define our performance metrics and then present our main results.

4.1 Performance Metrics

Any analysis of a data collection method must focus on latency and throughput. We make these notions precise in this section.

To address latency we first define a simpler data generation model: the *single round model*. In the single round model each node (apart from the sink) has a single data packet available at time $t = 0$. No further data appears. We define the *single round collection time* τ_{col} to be the number of time slots taken for all the data packets to reach the sink.

However, since the actual model we consider is of regular data generation at a rate β , we need a notion of latency in this setting as well. For this, let us identify different rounds of data generated. Starting time from $t = 0$, we number the data packets generated at any node starting from 1. We say that *round i* of data comprises all the $|V| - 1$ data packets that are numbered i .

The *data collection time* of the first k rounds of data, τ_{col}^k , is defined as,

$$\tau_{\text{col}}^k := \min_t \{X_{u_i, j}^t = u_s : 1 \leq j \leq k, u_i \in V \setminus u_s\} \quad (3)$$

where, u_s denotes the sink and $X_{u_i, j}^t$ is the random variable denoting the position of j^{th} round data packet of node u_i at the start of time slot t .

Definition 1 (Average data collection time). *The average data collection time for the network is defined by*

$$\bar{\tau} = \lim_{k \rightarrow \infty} \frac{\tau_{\text{col}}^k}{k}.$$

Turning to throughput we note that if the data arrival rate β is very high Metropolis-Collect will not be able to successfully move the data to the sink since the buffers will keep growing, i.e., the queues at the buffers will become unstable. So, first we define a *stable data rate* for any algorithm to

be a value of β that ensures $\bar{\tau} < \infty$, i.e. a data arrival rate is called stable if the average collection time is finite. We note that in Metropolis-Collect since no node is allowed to transmit more than 1 packet in any slot, all stable data rates satisfy $\beta < 1$. Our main theorem, Theorem 3, will give a lower bound on the stable data rate achievable by Metropolis-Collect and also give a general upper bound on stable data rates for any algorithm. As expected the notion of throughput is closely related to the notion stable data rate.

Definition 2 (Network throughput). *Given a stable data rate β , i.e. a data rate such that the average data collection time of Metropolis-Collect, $\bar{\tau}$, is finite, the network throughput is defined as the rate at which data is received by the sink. In other words, if we have m data sources ($m \leq n-1$) and a stable data rate β in a network, the network throughput is $m\beta$.*

4.2 Our Results

Theorem 1 (Single round data collection). *Given a graph $G = (V, E)$ with $|V| = n$ nodes, with maximum degree d_{\max} and minimum degree d_{\min} and worst-case hitting time of Metropolis random walk on G , $\max_{x, y \in V} \mathbb{E}_x(\tau_y)$ denoted by \hat{t}_{hit} , the single round data collection time τ_{col} for Metropolis-Collect is*

$$\tau_{\text{col}} = \left(4 \log n \hat{t}_{\text{hit}} + 2 \sqrt{30 \hat{t}_{\text{hit}} \log n} \right) = O(\log n \hat{t}_{\text{hit}}) \quad (4)$$

with probability at least $1 - 2/n$. Moreover $\hat{t}_{\text{hit}} \leq \frac{d_{\max}}{d_{\min}} t_{\text{hit}}$ where t_{hit} is the worst-case hitting time of simple random walk on G .

We note that t_{hit} is a natural lower bound on data collection in the sense that even if data had to be moved from just a single node to the sink, the time take would be t_{hit} . Theorem 1 shows that the time taken to collect a single round of data from all but one of the nodes of the network is just a logarithmic factor higher for d -regular networks or for networks that have low variability between the maximum and minimum degree. We present the proof of this theorem in detail in Section 5.

For the average data collection time we have the following result:

Theorem 2 (Average data collection time). *Given a graph G representing the underlying network where each node except the sink receives independent Bernoulli arrivals with rate β , then, if $\bar{\tau}$ is finite for Metropolis-Collect, i.e. β is a stable data rate then $\bar{\tau}$ is*

$$O\left(\frac{1}{\beta}\right) \quad (5)$$

with probability at least $1 - 2/n$.

The result of Theorem 2 is not surprising in the sense that since Metropolis-Collect works by ensuring that the expected buffer size is bounded, i.e. the queues are stable, we expect that the data being generated is cleared in a time inversely proportional to the rate in which it is generated. Detailed proof of Theorem 2 is presented in next section. Finally we turn to the main result on throughput.

Theorem 3 (Network throughput lower bound). *Given a graph $G = (V, E)$ with $|V| = n$ nodes and each node except the sink u_s having independent Bernoulli data arrivals with rate β , Metropolis-Collect is able to achieve a finite average data collection time $\bar{\tau}$ for β such that*

$$\beta \geq \frac{1 - \lambda_2}{\sqrt{n(n-1)}}$$

where λ_2 is the second largest eigenvalue of transition matrix \mathcal{P} of graph G with transition probability $\mathcal{P}[\cdot, \cdot]$ given by Metropolis-Collect. The corresponding network throughput is $(n-1)\beta$. In particular, for d -regular graphs

$$\frac{d - \lambda_2(A)}{d\sqrt{n(n-1)}} = \frac{1 - \lambda_2}{\sqrt{n(n-1)}} \leq \beta$$

where $\lambda_2(A)$ is the second largest eigenvalue of adjacency matrix of G .

The lower bound on throughput is shown to be related to the following natural upper bound on any data collection algorithm. In order to present this generalized upper bound, we first need to define a few terms. For any vertex subset $U \subset V$ we define its edge boundary as $\partial U := \{(u, v) : u \in U, v \notin U\}$. Now, for all $U \subset V$ we define a constant $\alpha(U) := \frac{|\partial U|}{|U|}$. And $\hat{\alpha}(G) := \min_{U \subset V, u_s \notin U} \alpha(U)$.

Note $\hat{\alpha}(G) \leq \alpha(G)$ where $\alpha(G) = \min_{U \subset V, |U| \leq \frac{1}{2}|V|} \frac{|\partial U|}{|U|}$ is edge expansion of graph G [11].

Proposition 1 (Generalized network throughput upper bound). *Given a graph $G = (V, E)$ with $|V| = n$ nodes and each node except the sink u_s having independent Bernoulli data arrivals with rate β , no data collection algorithm is able to achieve a finite average data collection time $\bar{\tau}$ for β such that*

$$\beta > \min \left\{ \hat{\alpha}(G), \frac{d_{u_s}}{n-1} \right\}$$

where d_{u_s} is the degree of the sink, $\hat{\alpha}(G) = \min_{U \subset V, u_s \notin U} \frac{|\partial U|}{|U|}$ is a constant and $\hat{\alpha}(G)$ is at most $\alpha(G)$, edge expansion of graph G .

5 Latency and throughput analysis

5.1 Latency analysis

We now present the proof of Theorem 1 that gives an upper bound on the time taken to collect a single round of data. This proof overcomes a significant difficulty: the fact that the Markov Chain induced by Metropolis-Collect on any data arrival pattern, even the single round arrival pattern where all data is available at $t = 0$ and no new data arrives, is *not* time homogeneous. The proof proceeds by first viewing each packets trajectory to the sink as a separate random walk. Now, given a packet p and let $X_{p,1}^t = X_p^t$ be its position at the start of time slot t . Consider the subset of time slots when Metropolis-Collect chooses p for transmission out of the buffer at node X_p^t , and the node actually moves (i.e. is not rejected by its neighbour). Let us call this subset $T(p)$. The key observation is that the process $\{X_p^t\}_{t \in T(p)}$ is a time homogeneous random walk on G . We first bound the time taken for $n-1$ such walks to reach the sink and then, in step 2, bound the delay between successive time steps in each $T(p)$ to reach the result.

Proof of Theorem 1. Let the hitting time of a random walk of a data packet from a node $u_i \in V \setminus u_s$ to the sink for Metropolis-Collect be defined as $\tau_i^s = \min_t \{X_{u_i}^t = u_s\}$. So, for a node $u_i \in V \setminus u_s$ we've $\mathbb{E}[\tau_i^s] \leq \hat{t}_{\text{hit}}$, where $\hat{t}_{\text{hit}} = \max_{x,y \in V} \mathbb{E}_x(\tau_y)$ is the worst-case hitting time of Metropolis random walk starting from any node of graph. By Markov's inequality $\mathbb{P}[\tau_i^s \geq 2\hat{t}_{\text{hit}}] \leq \frac{1}{2}$. The probability of a random walk not hitting the sink in w times the twice worst-case hitting time is $\mathbb{P}[\tau_i^s \geq w \cdot 2\hat{t}_{\text{hit}}] \leq \frac{1}{2^w}$. Now, for $w = 2 \log n$, we get

$$\mathbb{P}[\tau_i^s \geq 4 \log n \cdot \hat{t}_{\text{hit}}] \leq \frac{1}{n^2} \tag{6}$$

Thus, with probability at least $1 - 1/n^2$, the number of time slots taken by the data packet from the node u_i to hit the sink u_s are $4\hat{t}_{\text{hit}} \log n$. This analysis is done assuming there is only one packet in the queue of any node at any time.

Now, we analyze the queueing delay due to more than one packets in the queue $\mathcal{Q}_t(u)$. We will prove that no data packet is delayed by more than $2\sqrt{30\hat{t}_{\text{hit}}} \log n$ time slots with probability at least $1 - 1/n^2$ due to the queues at the nodes. Any packet X_j gets delayed at time slot t , due to queue at a node u_i , because it is not picked for transmission in that slot among all the packets in the queue. Thus, the probability of X_j being delayed by u_i at time t is,

$$\begin{aligned} \mathbb{P}[X_j \text{ is delayed}] &= \sum_{w=2}^n \frac{w-1}{w} \mathbb{P}[\mathcal{Q}_t(u_i) = w] \\ &= \frac{1}{2} \sum_{w=2}^n (w-1) \mathbb{P}[\mathcal{Q}_t(u_i) = w] \\ &\leq \frac{1}{2} \mathbb{E}[\mathcal{Q}_t(u_i)]. \end{aligned}$$

We know, initially every node except the sink has one data packet in its queue so, expected queue size $\mu_0(u_i) = \mathbb{E}[\mathcal{Q}_0(u_i)] \leq 1$. Now let us assume $\mu_t(u_i) = \mathbb{E}[\mathcal{Q}_t(u)] \leq 1$ holds true for any time slot t . We know that for Metropolis-Collect

$$\mu_{t+1}(u_i) = \mu_t(u_i) - \sum_{u_j: u_j \sim u_i} \mu_t(u_i) \mathcal{P}[u_i, u_j] + \sum_{u_j: u_j \sim u_i} \mu_t(u_j) \mathcal{P}[u_j, u_i]$$

And so, since the outgoing probability to any neighbour v , $\mathcal{P}[u, v]$, is equal to the incoming probability $\mathcal{P}[v, u] = 1/\max\{\deg(u), \deg(v)\}$, by induction we have $\mu_t(u_i) = \mathbb{E}[\mathcal{Q}_t(u_i)] \leq 1$. Hence,

$$\mathbb{P}[X_j \text{ is delayed}] \leq \frac{1}{2}. \quad (7)$$

Let H_j^m be a random variable which is 1 if packet X_j is delayed in time slot m and 0 otherwise. Then, $\mathbb{P}[H_j^m = 1] \leq \frac{1}{2}$. Let $Y_j^t = \sum_{m=1}^t H_j^m$ be a random variable which denotes the total delay incurred by a packet X_j in t time slots. So, $\mathbb{E}[Y_j^t] \leq \frac{t}{2}$. Let $Z_j^t := Y_j^t - \frac{t}{2}$. Then, $\mathbb{E}[Z_j^t] = \mathbb{E}[Y_j^t] - \frac{t}{2} \leq 0$. Since $Z_j^t = Z_j^{t-1} + H_j^t - \frac{1}{2}$ and $\mathbb{E}[Z_j^t | Z_j^{t-1}] = Z_j^{t-1} + \mathbb{E}[H_j^t] - \frac{1}{2} \leq Z_j^{t-1}$, we see that the sequence $\{Z_j^t\}_{t \geq 1}$ is a supermartingale.

Now, we know without any delays number of time slots taken by random walk of a data packet to hit the sink u_s is $4\hat{t}_{\text{hit}} \log n$ with probability at least $1 - 1/n^2$. Now, using Azuma's Inequality we show that the probability of exceeding that time by $2\sqrt{30\hat{t}_{\text{hit}}} \log n$ is very low. Let $t = 30\hat{t}_{\text{hit}} \log n$ then,

$$\begin{aligned} \mathbb{P}\left[Z_j^t \geq 2\sqrt{30\hat{t}_{\text{hit}}} \log n\right] &\leq \exp\left(-\frac{(2\sqrt{30\hat{t}_{\text{hit}}} \log n)^2}{2 \sum_{i=1}^t 1}\right) \\ &\leq \exp(-2 \log n) \leq \frac{1}{n^2} \end{aligned} \quad (8)$$

Thus, with probability at least $1 - 1/n^2$, $Z_j^t \leq 2\sqrt{30\hat{t}_{\text{hit}}} \log n \Rightarrow Y_j^t - \frac{t}{2} \leq 2\sqrt{30\hat{t}_{\text{hit}}} \log n$. So, $Y_j^t \leq 15\hat{t}_{\text{hit}} \log n + 2\sqrt{30\hat{t}_{\text{hit}}} \log n$. Let \bar{Y}_j^t represent the non-delayed slots so from Eq. (8) we have with probability at least $1 - 1/n^2$, $\bar{Y}_j^t \geq 15\hat{t}_{\text{hit}} \log n - 2\sqrt{30\hat{t}_{\text{hit}}} \log n = 4\hat{t}_{\text{hit}} \log n \left(\frac{15}{4} - \frac{1}{2} \sqrt{\frac{30}{\hat{t}_{\text{hit}}}}\right)$.

As, $\hat{t}_{\text{hit}} \geq 1$ this proves that none of the data packet's hitting event is delayed by more than $2\sqrt{30\hat{t}_{\text{hit}}} \log n$ time slots with probability at least $1 - 1/n^2$ due to the queues at each node.

Now, for the i^{th} data packet, we have *two events*: (1) $A_i := \{\tau_i^s \geq 4 \log n \hat{t}_{\text{hit}}\}$ and (2) $B_i := \{Z_j^t \geq 2\sqrt{30\hat{t}_{\text{hit}}} \log n\}$. We have shown that $\mathbb{P}[A_i] \leq 1/n^2$ and $\mathbb{P}[B_i] \leq 1/n^2$. This means that $\mathbb{P}[A_i \cup B_i] \leq 2/n^2$.

So, for all n data packets we need to find $\bigcup_{i=1}^n A_i \cup B_i$. So,

$$\mathbb{P}\left[\bigcup_{i=1}^n A_i \cup B_i\right] \leq \frac{2}{n}. \quad (9)$$

So, from equations (6) and (8), all n walks take $4 \log n \hat{t}_{\text{hit}} + 2\sqrt{30\hat{t}_{\text{hit}}} \log n$ time slots to hit the sink with probability at least $1 - 2/n$.

Moreover, since the minimum acceptance probability of any transition in the Metropolis Chain is $\frac{d_{\min}}{d_{\max}}$, the expected holding time at any state due to rejections is at most $\frac{d_{\max}}{d_{\min}}$ and so $\hat{t}_{\text{hit}} \leq \frac{d_{\max}}{d_{\min}} t_{\text{hit}}$ where t_{hit} is the worst-case hitting time of simple random walk on G . This proves the theorem. \square

5.2 Average Data Collection Time

Consider our network setting where we have independent Bernoulli data arrivals with stable rate β given by Theorem 3 and Proposition 1 which ensure finite average data collection time $\bar{\tau}$. For each round of data arrival coming at the rate β we have n random walks of data packets moving on graph G representing the underlying network. Now we find the average data collection time for k such rounds of data arrival.

Proof of Theorem 2. For a given graph G where each node except the sink receives independent Bernoulli arrivals with stable rate β (see Theorem 3 and Proposition 1) which ensures finite average data collection time $\bar{\tau}$. Recall each data round generated has total of $|V| - 1$ data packets and each round has its appearance and clearance time in the network. Instead of finding such times individually for each round, we will proceed our analysis by finding the maximum time by which k rounds of data arrival have happened and then after this time, we find the clearance time assuming all nk packets have appeared.

Let τ_{app}^k be the appearance time of k rounds of data arrival on each node using Metropolis-Collect algorithm for communication. Let f be a hypothetical node where we assume packets reside before arriving at the network nodes, then, $\forall u_i \in V$ and $1 \leq j \leq k$, we have $X_{u_i,j}^0 = f$. If the j^{th} data item appears at node u_i at time t' then $X_{u_i,j}^t = f$ for all $t < t'$ and $X_{u_i,j}^{t'} = u_i$. With this notation we can define appearance time as: $\tau_{\text{app}}^k = \min_t \{X_{u_i,j}^t \in V : 1 \leq j \leq k, u_i \in V\}$. So, τ_{app}^k is the earliest time when k packets have appeared at each node. Now, we prove a high probability bound on the appearance time τ_{app}^k .

Let A be the event that a source node $u_i \in V$ did not receive k arrivals in time t . Consider, $t \geq 2k$ and $\beta \leq 1/2$. So, we have

$$\mathbb{P}[A] = \sum_{i=1}^k \binom{t}{k-i} \beta^{k-i} (1-\beta)^{t-(k-i)} \quad (10)$$

As, $t \geq 2k$, $\binom{t}{k-i} \leq \binom{t}{k-(i-1)} \forall i : 1 \leq i-1 \leq k$ also, $\beta \leq 1/2$, so Equation (10) can be written as,

$$\mathbb{P}[A] \leq k \binom{t}{k} (1-\beta)^t$$

Using, $\binom{t}{k} \leq (\frac{et}{k})^k$ and let $t = ck$ for $c > 1$, above equation can be rewritten as,

$$\mathbb{P}[A] \leq k(ec(1-\beta)^c)^k$$

So, we can easily find c such that $ec(1-\beta)^c \leq 2/3$. Note that the value of c will be only β dependent as $c = 1 + \frac{b}{\beta}$ for some constant $b > 1$ (for finding such value of c see Lemma 3 of Appendix A). So, for a node $u_i \in V$,

$$\mathbb{P}[t \geq ck] \leq k \left(\frac{2}{3}\right)^k$$

Since, all random walks of data packets from all n nodes are independent of each other, considering worse case analysis of all nodes we have,

$$\mathbb{P}[\exists i : t_i \geq ck] \leq nk \left(\frac{2}{3}\right)^k \quad (11)$$

So, from Equation (11) with high probability the appearance time of k rounds of data arrival in a stable network is at most ck where $c = 1 + \frac{b}{\beta}$ for some constant $b > 1$, so,

$$\tau_{\text{app}}^k \leq \left(1 + \frac{b}{\beta}\right)k \quad (12)$$

Note that after τ_{app}^k time all nodes have k arrivals. Let $\tau_{\text{clr}}(k)$ be the clearance time of random walks when k data packets have arrived at all the nodes. We know from Theorem 1 the clearance time of random walks assuming each node has only one data packet is $\tau_{\text{clr}}(1) = \tau_{\text{col}} = \left(4\frac{d_{\text{max}}}{d_{\text{min}}} \log n t_{\text{hit}} + 2\sqrt{30\frac{d_{\text{max}}}{d_{\text{min}}} t_{\text{hit}} \log n}\right)$. Also, because of the appearance time of rounds, random walks across the data rounds are independent, so with probability at least $1 - 2/n$,

$$\tau_{\text{clr}}(k) \leq k\tau_{\text{clr}}(1) = k \left(4\frac{d_{\text{max}}}{d_{\text{min}}} \log n t_{\text{hit}} + 2\sqrt{30\frac{d_{\text{max}}}{d_{\text{min}}} t_{\text{hit}} \log n}\right). \quad (13)$$

Recall, τ_{col}^k is the collection time of k rounds of data arrival and so, we can write

$$\tau_{\text{col}}^k = \tau_{\text{app}}^k + \tau_{\text{clr}}(k). \quad (14)$$

Now, using the results from Eq.s (12) and (13), we have

$$\tau_{\text{col}}^k \leq \left(1 + \frac{b}{\beta}\right)k + k \left(4\frac{d_{\text{max}}}{d_{\text{min}}} \log n t_{\text{hit}} + 2\sqrt{30\frac{d_{\text{max}}}{d_{\text{min}}} t_{\text{hit}} \log n}\right) \quad (15)$$

So, with probability at least $1 - 2/n$ the average data collection time of k rounds is,

$$\bar{\tau} = \lim_{k \rightarrow \infty} \frac{\tau_{\text{col}}^k}{k} \leq \left(1 + \frac{b}{\beta}\right) + \left(4\frac{d_{\text{max}}}{d_{\text{min}}} \log n t_{\text{hit}} + 2\sqrt{30\frac{d_{\text{max}}}{d_{\text{min}}} t_{\text{hit}} \log n}\right) \quad (16)$$

Let t_{cov} be the time required by a random walk to cover all the states (see Chapter 11 [28]). Recall the definition of t_{hit} . Let $x, y \in V$ be states for which $t_{\text{hit}} = \mathbb{E}_x \tau_y$, thus any walk starting at x must have visited y by the time all states are covered, so we have $t_{\text{hit}} \leq t_{\text{cov}}$. For a connected graph like the given graph G it is not possible for the random walk to assign non-zero probability to a vertex it has not yet visited, so we can conclude that $t_{\text{cov}} \leq t_{\text{mix}}$, where t_{mix} is the mixing time of graph (see Section 4.5 [28]). We know $\beta \geq \frac{1-\lambda_2}{\sqrt{n(n-1)}}$ (by Theorem 3) and $t_{\text{hit}} \leq t_{\text{mix}} \leq \frac{\log n}{1-\lambda_2}$ (Theorem 12.3 [28]). Using above results in Eq. (16) we prove Theorem 2. \square

5.3 Data Rate Bounds

In the previous section, we proved latency bounds for Metropolis-Collect where we considered that each node except the sink receives data at rate β . So, for our setting recall from Section 4.1 to successfully collect all data we need to ensure this rate is stable i.e. average collection time $\bar{\tau}$ is finite. So, in this section, we prove bounds on such rate and the corresponding throughput as two are closely related. In particular, we prove the lower bound for Metropolis-Collect and a general upper bound for any data collection algorithm. We also generalize the results for m data sources and find a sufficient bound on rate.

Proof of Theorem 3. For our given graph $G = (V, E)$ with n nodes out of which one is sink u_s , let data arrive as independent Bernoulli arrivals with rate β at the source nodes (all nodes except sink). Now, for Metropolis-Collect we find the bound for stable data rate which ensures finite average collection time $\bar{\tau}$ such that it successfully collect all data.

For the data arrival rate β at each node (except sink) and transition matrix \mathcal{P} of graph G for the Metropolis-Collect algorithm, let μ be an n element row vector representing the steady state queues of nodes (as discussed in Section 3.3) in graph i.e. for nodes $u_1, u_2, \dots, u_{n-1}, u_s$ we have, $\mu = [\mu_1 \ \mu_2 \ \dots \ \mu_{n-1} \ 0]$ respectively. This is defined assuming that sink collects all data it receives and has no notion of maintaining queue. Let e_n be another n element row vector for nodes in graph such that $e_n = [0 \ 0 \ \dots \ 0 \ 1]$. Let I be the usual $n \times n$ identity matrix and $J = [1 \ 1 \ \dots \ 1 \ 1]$ be an n element all one row vector. The steady state queue equations at nodes can be written in vector form as

$$\begin{aligned} \mu \mathcal{P} + \beta(J - ne_n) &= \mu \\ \mu(I - \mathcal{P}) &= \beta(J - ne_n) \end{aligned} \tag{17}$$

Let π be a distribution on V satisfying $\pi = \pi \mathcal{P}$, then π is said to be a stationary distribution. For the purpose of this proof we consider π to be uniform i.e. $\pi(u) = \frac{1}{|V|}$ for every $u \in V$. Let the transition matrix \mathcal{P} be reversible with respect to the stationary distribution π i.e. $\pi(x)\mathcal{P}(x, y) = \pi(y)\mathcal{P}(y, x), \forall x, y \in V$. The usual inner product on the vector space \mathbb{R}^V is given by $\langle f, g \rangle = \sum_{x \in V} f(x)g(x)$. We define another inner product on \mathbb{R}^V that we will use in this proof $\langle f, g \rangle_\pi := \sum_{x \in V} f(x)g(x)\pi(x)$. From Lemma 12.2 [28], the inner product space $(\mathbb{R}^V, \langle \cdot, \cdot \rangle_\pi)$ has an orthonormal basis of real-valued eigenfunctions $\{f_j\}_{j=1}^{|V|}$ corresponding to real eigenvalues $\{\lambda_j\}$. Writing the vector μ in terms of the eigenvectors, we have $\mu = \sum_{i=1}^{|V|} \langle \mu, f_i \rangle_\pi f_i$. This gives us that $\mu(I - \mathcal{P}) = \sum_{i=1}^{|V|} (1 - \lambda_i) \langle \mu, f_i \rangle_\pi f_i$. From the Perron-Frobenius theorem (see e.g. [37, Chap 9]) and Lemma 12.1 of [28], we know $\lambda_1 = 1 > \lambda_2 \geq \dots \geq \lambda_n > -1$. So, we have

$$\mu(I - \mathcal{P}) = \sum_{i=2}^{|V|} (1 - \lambda_i) \langle \mu, f_i \rangle_\pi f_i \tag{18}$$

$$\geq (1 - \lambda_2) \left(\sum_{i=2}^{|V|} \langle \mu, f_i \rangle_\pi f_i \right) \tag{19}$$

Note, that $f_1, \dots, f_{|V|}$ form an orthonormal basis so, $\sum_{i=1}^{|V|} \langle \mu, f_i \rangle_\pi^2 = \|\mu\|_\pi^2$. Hence we have

$$\sum_{i=2}^n \langle \mu, f_i \rangle_\pi^2 = \|\mu\|_\pi^2 - \langle \mu, f_1 \rangle_\pi^2 \tag{20}$$

The eigenfunction f_1 corresponding to the eigenvalue 1 can be taken to be a constant vector $\mathbf{1}$ (see Lemma 12.2 [28]), so $\langle \mu, f_1 \rangle_\pi = \sum_{i=1}^n \mu_i/n$. Also, $\|\mu\|_\pi^2 = \sum_{i=1}^n \mu_i^2/n$. So, using these results in Eq. (20) we have

$$\sum_{i=2}^n \langle \mu, f_i \rangle_\pi^2 = \sum_{i=1}^n \frac{\mu_i^2}{n} - \left(\sum_{i=1}^n \frac{\mu_i}{n} \right)^2 = \text{Var}(\mu_i) = \sum_{i=1}^n \frac{(\mu_i - \bar{\mu})^2}{n} \quad (21)$$

where, $\bar{\mu}$ is the expected queue size. Now, taking the square of norm of Eq. (19) and using Eq. (20), we have

$$\|\mu(I - \mathcal{P})\|_\pi^2 \geq (1 - \lambda_2)^2 \left(\sum_{i=1}^n \frac{\mu_i^2}{n} - \left(\sum_{i=1}^n \frac{\mu_i}{n} \right)^2 \right) \quad (22)$$

Also,

$$\|\beta(J - ne_n)\|_\pi^2 = \frac{\beta^2}{n} (n - 1 + (n - 1)^2) = \beta^2(n - 1) \quad (23)$$

Now, again taking the square of norm of Eq. (17) and using Eq.s (22) and (23), we have

$$\beta^2(n - 1) \geq (1 - \lambda_2)^2 \left(\sum_{i=1}^n \frac{\mu_i^2}{n} - \left(\sum_{i=1}^n \frac{\mu_i}{n} \right)^2 \right) \quad (24)$$

To get a lower bound for β we need a lower bound on $\text{Var}(\mu_i)$. For this we first consider two nodes whose queue size we know precisely (1) the sink, u_s , which has $\mu_s = 0$, and (2) a node v_{\max} , such that $\mu_{v_{\max}} = 1$. While we don't know precisely which node is v_{\max} , we know that such a node exists, because if it didn't then we could always raise the value of β , i.e. the fact that β is maximum implies that some node's expected queue size is 1. We note that the contribution of s and v_{\max} to the RHS of ((24)) is

$$\frac{(1 - \bar{\mu})^2 + (\bar{\mu} - 0)^2}{n} \geq \frac{1}{2n}, \quad (25)$$

since for any $x \in [0, 1]$, $x^2 + (1 - x)^2$ takes a minimum value of $1/2$ at $x = 1/2$.

Noting that, since the arrival rate β is stable, at steady state the expected number of packets entering the sink at any time is $\sum_{u \sim s} \mu(u) \mathcal{P}[u, s] = (n - 1)\beta$, and observing that $\sum_{u \sim s} \mathcal{P}[u, s] \leq 1$, we conclude that the sink has at least one neighbour, call it s' , that has $\mu_{s'} \geq (n - 1)\beta$. Let us observe here that $\bar{\mu} = (n - 1)\beta/n$ since the system is stable and so at any point in time after steady state is reached the total number of packets that have been generated but not yet sunk is $(n - 1)\beta$ in expectation. Hence we know that $(n - 1)\beta \geq \bar{\mu}$ and so, $\mu_{s'} - \bar{\mu} \geq (n - 1)\beta - \bar{\mu}$.

Similarly, since the expected number of packets coming into v_{\max} is exactly $1 - \beta$ at steady state, and since $\sum_{v \sim v_{\max}} \mathcal{P}[v, v_{\max}] \leq 1$, there is a neighbour of v_{\max} , we call it m' , such that $\mu(m') \geq 1 - \beta$. Now, noting that since the average rate at which the sink can take in packets is at most 1 (because $\sum_{u \sim s} \mathcal{P}[u, s] \leq 1$), therefore $\beta < 1/(n - 1)$. This means that $1 - \beta \geq (n - 2)/(n - 1)$ which is larger than $\bar{\mu} = (n - 1)\beta/n$. So, we can say that $\mu_{m'} - \bar{\mu} \geq (1 - \beta) - \bar{\mu}$. So, in order to derive a lower bound on the contribution of s' and m' to the RHS of (24) we need to minimize $(\bar{\mu} - (1 - \beta))^2 + (\bar{\mu} - (n - 1)\beta)^2$. Using calculus and the fact that $0 \leq \beta \leq 1/(n - 1)$ we get

$$(\bar{\mu} - (1 - \beta))^2 + (\bar{\mu} - (n - 1)\beta)^2 \geq \frac{1}{2} \quad (26)$$

Putting (25) and (26) back into (24), we get

$$\sum_{i=1}^n \frac{(\mu_i - \bar{\mu})^2}{n} \geq \frac{1}{n}. \quad (27)$$

Using the variance bound (27) in Eq. (24), we have the lower bound on rate as

$$\beta \geq \frac{1 - \lambda_2}{\sqrt{n(n-1)}}. \quad (28)$$

For d -regular graph, spectral gap is $d - \lambda_2(A)$ where $\lambda_2(A)$ is the second largest eigenvalue of its adjacency matrix [11]. Also, it is easy to check that $\lambda_2(A) = d\lambda_2$ where λ_2 is the corresponding eigenvalue of transition matrix \mathcal{P} of random walk on given graph for Metropolis-Collect. So, for d -regular graphs in particular we have,

$$\beta \geq \frac{1 - \lambda_2}{\sqrt{n(n-1)}} = \frac{d - \lambda_2(A)}{d\sqrt{n(n-1)}} \quad (29)$$

This proves the lower bound on rate β . Since, the network throughput is the rate at which the sink receives the data, so for our network with $n - 1$ data sources each receiving data at stable rate β , corresponding network throughput is $(n - 1)\beta$. \square

For our given graph $G = (V, E)$ with n nodes, with each node except the sink u_s receiving data as independent Bernoulli arrivals with rate β , we find the maximum value of data arrival rate β such that any data collection algorithm can achieve a finite average data collection time $\bar{\tau}$.

Proof of Proposition 1. We know, for any data collection algorithm given a set $U \subset V$, where $u_s \notin U$ the maximum data flow that can move out of this set is the flow across the boundary,

$$\beta|U| \leq |\partial U| \quad (30)$$

$$\beta \leq \min_U \alpha(U) = \hat{\alpha}(G) \leq \alpha(G) \quad (31)$$

Also, for set $U = V \setminus u_s$ we have $\alpha(U) = \frac{d_{u_s}}{n-1}$, where d_{u_s} is the degree of the sink. Now, from Eq. 30 $\beta \leq \frac{d_{u_s}}{n-1}$. So, for any data collection algorithm upper bound on stable rate β is given by

$$\min \left\{ \hat{\alpha}(G), \frac{d_{u_s}}{n-1} \right\} \quad (32)$$

\square

Discussion about network throughput upper bound for Metropolis-Collect In particular, if we consider Metropolis-Collect algorithm with transition matrix \mathcal{P} . In this algorithm, instead of deterministically sending a data packet along any edge we send it with some probability given by $\mathcal{P}[\cdot, \cdot]$. Now, for any vertex $v \in V$, we define its measure as, $\rho(v) := \sum_{u \in V} \mathcal{P}[u, v]$. Similarly, for any $U \subset V$ we define the measure $\rho(U) = \sum_{v \in U} \rho(v)$ and we define its edge boundary as $\partial U := \{(u, v) : u \in U, v \notin U\}$. Thus, $\rho(\partial U) = \sum_{u \in U, v \notin U} \mathcal{P}[u, v]$. Now, we define constants $h(U) := \frac{\rho(\partial U)}{\rho(U)}$ and $\hat{h}(G) := \min_{U \subset V, u_s \notin U} h(U) \leq h(G)$ where $h(G)$ is the *Cheeger's constant* for the random walk on the graph G . Now, for $U = V \setminus u_s$,

$$\hat{h}(G) \leq \sum_{u \sim u_s} \frac{\mathcal{P}[u, u_s]}{n-1} \quad (33)$$

We know, for any given set $U \subset V$, where $u_s \notin U$ the maximum data flow that can move out of this set is the flow across the boundary,

$$\beta \rho(U) \leq \rho(\partial U) \quad (34)$$

$$\beta \leq \min_U h(U) = \hat{h}(G) \leq h(G) \quad (35)$$

From Eq. (34), flow out of set U can also be written as,

$$\beta|U| \leq h(U)|U| \quad (36)$$

The fundamental necessary bound on the value of β is the accepting rate of the sink, so we have $\beta(n-1-|U|) + \beta|U| \leq \mathcal{P}[u, u_s]$. Using Eq. (36) in the above equation we have,

$$\begin{aligned} \beta(n-1-|U|) + h(U)|U| &\leq \mathcal{P}[u, u_s] \\ (n-1)\beta &\leq \mathcal{P}[u, u_s] - (h(U) - \beta)|U| \end{aligned} \quad (37)$$

Using Eq. (34) in Eq. (37), we have $(h(U) - \beta) \geq 0$, so

$$\beta \leq \sum_{u \sim u_s} \frac{\mathcal{P}[u, u_s]}{n-1} \quad (38)$$

So, the bottleneck for rate can be sink itself or some other far-off node. Hence, the maximum value of stable data rate i.e. rate ensuring finite average data collection time $\bar{\tau}$ for Metropolis-Collect is given by,

$$\beta \leq \min \left\{ \hat{h}(G), \sum_{u \sim u_s} \frac{\mathcal{P}[u, u_s]}{n-1} \right\} \quad (39)$$

Now, if we compare the general upper bound for any data collection algorithm (Eq. (32)) with Metropolis-Collect's upper bound (Eq. (39)), since $\sum_{u \sim u_s} \mathcal{P}[u, u_s] = 1$ and for regular graphs $\hat{h}(G) = \frac{\hat{\alpha}(G)}{d}$, so we always achieve a rate which is at least a factor d less than any other data collection algorithm. This result is clear from the simulations (Figure 3) where we compare Metropolis-Collect with Directed diffusion method. Similarly, for non-regular graphs since $\sum_{u \sim u_s} \mathcal{P}[u, u_s] \leq 1$ and $\hat{h}(G) \leq \frac{\hat{\alpha}(G)}{d_{min}}$ where d_{min} is the minimum degree of the graph G , so we are at least a factor d_{min} less than others. So, in order to achieve data collection in low configured networks using Metropolis-Collect we need to compromise on rate by certain factor.

Next, we generalize our results for a network of n nodes with only m source nodes, one sink and rest as relay nodes.

Lemma 1. *For a given graph $\bar{G} = (\bar{V}, \bar{E})$ with $|\bar{V}| = n$ nodes and source set $V_s \subset \bar{V}$ with $|V_s| = m$ data sources, each receiving data as independent Bernoulli arrivals with stable data arrival rate β (which ensures finite average data collection time $\bar{\tau}$) is given by*

$$\beta \geq \frac{1 - \lambda_2}{\sqrt{m(m+1)}} \quad (40)$$

where λ_2 is the second largest eigenvalue of transition matrix $\bar{\mathcal{P}}$ of graph \bar{G} given by Metropolis-Collect. The corresponding network throughput is $(n-1)\beta$. In particular, for d -regular graphs

$$\frac{d - \lambda_2(\bar{A})}{d\sqrt{m(m+1)}} = \frac{1 - \lambda_2}{\sqrt{m(m+1)}} \leq \beta$$

where $\lambda_2(\bar{A})$ is the second largest eigenvalue of adjacency matrix of \bar{G} .

Proof. For a given graph $\bar{G} = (\bar{V}, \bar{E})$, let, the source set be $V_s \subset \bar{V}$ such that $|V_s| = m$, β be the data arrival rate and $\bar{\mathcal{P}}$ be the transition matrix of given graph for the Metropolis-Collect algorithm. We consider steady state queue vector μ and e_n vector same as defined in proof of Theorem 3. Let, I be the usual identity matrix, so only row vector J needs to be redefined. Let the n element row vector J be defined as follows

$$J_i := \begin{cases} 1 & \forall i \in V_s \\ 0 & \text{otherwise} \end{cases}$$

So, steady state queue equations at nodes is given by,

$$\begin{aligned} \mu \bar{\mathcal{P}} + \beta(J - me_n) &= \mu \\ \mu(I - \bar{\mathcal{P}}) &= \beta(J - me_n) \end{aligned} \quad (41)$$

Now, using the analysis similar to Step 1 of the proof of Theorem 3, we have

$$\|\mu(I - \bar{\mathcal{P}})\|_\pi^2 \geq (1 - \lambda_2)^2 \left(\sum_{i=1}^n \frac{\mu_i^2}{n} - \left(\sum_{i=1}^n \frac{\mu_i}{n} \right)^2 \right) \quad (42)$$

But, now based on new J , we have

$$\|\beta(J - me_n)\|_\pi^2 = \frac{\beta^2}{n} (m + m^2) = \frac{\beta^2}{n} m(m + 1) \quad (43)$$

Using variance bound (Eq. (27)) and Eq.s (42) and (43) in Eq. (41), we have the lower bound on stable rate which ensures finite average data collection time $\bar{\tau}$,

$$\beta \geq \frac{1 - \lambda_2}{\sqrt{m(m + 1)}} \quad (44)$$

For d -regular graph, we have $\lambda_2(\bar{A}) = d\lambda_2$ where $\lambda_2(\bar{A})$ is the second largest eigenvalue of its adjacency matrix \bar{A} and λ_2 is the corresponding eigenvalue of transition matrix $\bar{\mathcal{P}}$ of random walk on given graph for Metropolis-Collect. So, similar to Theorem 3 proof spectral gap for graph is $d - \lambda_2(\bar{A})$. So, we can write,

$$\beta \geq \frac{1 - \lambda_2}{\sqrt{m(m + 1)}} = \frac{d - \lambda_2(\bar{A})}{d\sqrt{m(m + 1)}}$$

Hence, the corresponding network throughput is $m\beta$. \square

Similar to Eq. (39), for graph \bar{G} with m data sources upper bound on stable rate β for Metropolis-Collect with transition matrix $\bar{\mathcal{P}}$ is given by

$$\beta \leq \min \left\{ \hat{h}(\bar{G}), \sum_{u \sim u_s} \frac{\bar{\mathcal{P}}\{u, u_s\}}{m} \right\}.$$

Next, we find a sufficient bound on rate i.e. $\forall \beta_i < \beta$ network stability is ensured, in terms of basic graph parameters. Since, in our network at every time slot, every node except the sink keeps on receiving and forwarding data to their neighbors whereas, sink keeps on receiving data but does not transmit any data towards its neighbors, thereby creating some space in queues of such nodes. This created space propagates among the other nodes and thus can be exploited to receive new arrivals at the nodes. Proof of this lemma is done by analyzing such spaces in queues based on expected queue equation at each node using Breadth-first search analysis (BFS) where interconnection between nodes is ignored.

Lemma 2. For a given graph G with data arriving at nodes with rate β , sufficient bound on rate which ensures network stability such that the average collection time $\bar{\tau}$ is finite, is given by,

$$\beta \leq \frac{d_{\min} - 1}{d_{\min} d_{\max}^D} \left(1 - \frac{1}{d_{\min}^D}\right)^{-1} \quad (45)$$

where d_{\min} , d_{\max} are minimum and maximum degrees of nodes and D is the diameter of graph G .

Proof. For our given graph G at every time step $t + 1$, with new arrivals coming at rate β queues at each node $u_i \in V \setminus u_s$ is updated as follows,

$$\mu_{t+1}(u_i) = \mu_t(u_i) - \sum_{u_j: u_j \sim u_i} \mu_t(u_i) \mathcal{P}[u_i, u_j] + \sum_{u_j: u_j \sim u_i} \mu_t(u_j) \mathcal{P}[u_j, u_i] + \beta \quad (46)$$

We know, for analysis of general graphs we maintain bounded queue size at each node $u_i \in V \setminus u_s$ i.e. $\mu_t(u_i) \leq 1$. Suppose, $\mu_t(u_i) = 1 - \epsilon$ where, $\epsilon > 0$ then, for $\mu_t(u_i) \leq 1$, β should be strictly less than ϵ to ensure bounded expected queue size constraint. Now, to find the arrival rate β such that stability of network i.e. finite average data collection time is ensured we will analyse the space created by sink at the queues of its neighbors which is eventually propagated to the queues of subsequent nodes. So, Eq.(46) can be written as,

$$\mu_{t+1}(u_i) \leq 1 - \sum_{u_j: u_j \sim u_i} \mathcal{P}[u_i, u_j] + \sum_{u_j: u_j \sim u_i} \mathcal{P}[u_j, u_i] + \beta \quad (47)$$

Now, for level 1 nodes (nodes at the distance one from sink) Eq.(47) can be written as,

$$\begin{aligned} \mu_t(u_1) &\leq 1 - \sum_{v: v \sim u_1} \mathcal{P}[u_1, v] + \sum_{v: v \sim u_1} \mathcal{P}[v, u_1] - \mathcal{P}[u_s, u_1] + \beta \\ &\leq 1 - \mathcal{P}[u_s, u_1] + \beta \end{aligned} \quad (48)$$

As sink does not forward any data to level 1 node, we subtract the probability of sink sending data to level 1 node. For level 2 nodes we have,

$$\begin{aligned} \mu_t(u_2) &\leq 1 - \sum_{v: v \sim u_2} \mathcal{P}[u_2, v] + \sum_{v: v \sim u_2} \mathcal{P}[v, u_2] + \beta \\ &\leq 1 - \sum_{v: v \sim u_2} \mathcal{P}[u_2, v] + \sum_{v: v \sim u_2} \mathcal{P}[v, u_2] - \mathcal{P}[u_1, u_2] \\ &\quad + \mathcal{P}[u_1, u_2](1 - \mathcal{P}[u_s, u_1] + \beta) + \beta \\ &\leq 1 - \mathcal{P}[u_1, u_2] + \mathcal{P}[u_1, u_2](1 - \mathcal{P}[u_s, u_1] + \beta) \\ &\leq 1 - \mathcal{P}[u_1, u_2]\mathcal{P}[u_s, u_1] + \beta\mathcal{P}[u_1, u_2] + \beta \end{aligned}$$

In the above equation, we first subtract the contribution to arrival rate by node u_1 with queue size less than one then we add its contribution with its actual queue size as given by (48).

For level 3 nodes we have,

$$\begin{aligned} \mu_t(u_3) &\leq 1 - \sum_{v: v \sim u_3} \mathcal{P}[u_3, v] + \sum_{v: v \sim u_3} \mathcal{P}[v, u_3] + \beta \\ &\leq 1 - \mathcal{P}[u_2, u_3]\mathcal{P}[u_1, u_2]\mathcal{P}[u_s, u_1] + \beta(1 + \mathcal{P}[u_2, u_3] + \mathcal{P}[u_2, u_3]\mathcal{P}[u_1, u_2]) \end{aligned}$$

Similarly, for nodes at level D we have,

$$\begin{aligned}\mu_t(u_D) &\leq 1 - (\mathcal{P}[u_{D-1}, u_D] \cdots \mathcal{P}[u_s, u_1]) \\ &\quad + \beta \left(1 + \mathcal{P}[u_{D-1}, u_D] + \mathcal{P}[u_{D-1}, u_D] \mathcal{P}[u_{D-2}, u_{D-1}] \right. \\ &\quad \left. + \cdots + (\mathcal{P}[u_{D-1}, u_D] \cdots \mathcal{P}[u_1, u_2]) \right)\end{aligned}\tag{49}$$

Now, we know $\mu_t(u_D) \leq 1$ so using Eq. (49) we get,

$$\begin{aligned}1 - (\mathcal{P}[u_{D-1}, u_D] \cdots \mathcal{P}[u_s, u_1]) + \beta \left(1 + \mathcal{P}[u_{D-1}, u_D] + \cdots \right. \\ \left. + (\mathcal{P}[u_{D-1}, u_D] \cdots \mathcal{P}[u_1, u_2]) \right) \leq 1 \\ \beta \leq \frac{(\mathcal{P}[u_{D-1}, u_D] \cdots \mathcal{P}[u_s, u_1])}{\left(1 + \mathcal{P}[u_{D-1}, u_D] + \cdots + (\mathcal{P}[u_{D-1}, u_D] \cdots \mathcal{P}[u_1, u_2]) \right)}\end{aligned}$$

For our modified random walk, we've $\mathcal{P}[u_i, u_j] = \frac{1}{\deg(u_i)} \left[\frac{\deg(u_i)}{\deg(u_j)} \wedge 1 \right]$. So, to get a sufficient bound on β we have,

$$\begin{aligned}\beta &\leq \frac{1}{d_{\max}^D} \cdot \frac{1}{\sum_{i=0}^{D-1} 1/d_{\min}^i} \\ &\leq \frac{d_{\min} - 1}{d_{\min} d_{\max}^D} \left(1 - \frac{1}{d_{\min}^D} \right)^{-1}\end{aligned}$$

In worse case, the distance from sink D can be the diameter of graph, hence the result. \square

5.4 Discussion

We find that the rate at which Metropolis-Collect algorithm can collect data is lower bounded by the spectral gap of random walk's transition matrix which is proportional to the spectral gap of the network for regular graphs. The rate for any data collection algorithm is upper bounded by the edge expansion of graph or the accepting rate of sink depending on whichever is minimum. We also find the corresponding network throughput with a stable data arrival rate β which falls within these bounds. The cost incurred to achieve such throughput in a *zero configuration* operation mode is not too high. From Theorem 1 we can see that our latency for a single round of data collection is just logarithmic factor higher than worst case latency t_{hit} , as we have bounded degree graphs for practical networks. Also, from the proof of Theorem 2 we get that as the number of rounds increase, their average collection time depends only on (inversely proportional to) data arrival rate β . This is evident from the fact that, once β is fixed, network stability is ensured, so data keeps on flowing freely in the network and gets successfully collected at the sink. So, the only dependence for such collection time will be the appearance time factor β .

5.5 Examples

5.5.1 Cycle graph

Let G be a n node even cycle or ring graph. So we know, for the given graph second largest eigenvalue of transition matrix for Metropolis-Collect is $\lambda_2 = 1 - O(1/n^2)$ (see Section 12.3 [28],[36]). So, the lower bound on stable rate given by Theorem 3 is $\beta \geq \frac{1-\lambda_2}{\sqrt{n(n-1)}} = O\left(\frac{1}{n^3}\right)$

Table 1: Rate bounds for various graphs

Graph	Lower bound	Upper bound
Cycle	$O\left(\frac{1}{n^3}\right)$	$\frac{1}{n-1}$
Star Graph with sink at center	$\frac{1}{\sqrt{n(n-1)^3}}$	$\frac{1}{n-1}$
Random Geometric Graph	$O\left(\frac{\log n}{n^2}\right)$	$O\left(\frac{1}{n-1}\right)$

Now, for cycle graph we have, for all nodes $u_i, u_j \in V$, $\mathcal{P}[u_i, u_j] = \mathcal{P}[u_j, u_i] = 1/2$, so $\sum_{u \sim u_s} \mathcal{P}[u, u_s] = 1$. Also, vertex set $U = V \setminus u_s$ minimizes the value of $h(U)$, so $\hat{h}(G) = \sum_{u \sim u_s} \mathcal{P}[u, u_s]/n - 1$ for any $u \in U$. So, from Eq. (39) upper bound on stable rate is $\beta \leq \sum_{u \sim u_s} \frac{\mathcal{P}[u, u_s]}{n-1} = \frac{1}{n-1}$.

To validate our bounds we find the stable data rate using first principles. Using steady state queue equation (17) for immediate neighbors of sink u_s we have, for node u_1 : $-\mu_1 + \frac{\mu_2}{2} + \beta = 0 \Rightarrow \mu_2 - \mu_1 = \mu_1 - 2\beta$ and for node u_{n-1} : $-\mu_{n-1} + \frac{\mu_{n-2}}{2} + \beta = 0 \Rightarrow \mu_{n-2} - \mu_{n-1} = \mu_{n-1} - 2\beta$. Similarly, for any general node u_i which is not a neighbor of sink (17) gives $(\mu_{i+1} - \mu_i) = (\mu_i - \mu_{i-1}) - 2\beta$. Using steady state equations of node u_1 , u_{n-1} and general node $u_i \in V \setminus \{u_1, u_{n-1}, u_s\}$, we get

$$\mu_1 = 2(n-1)\beta - \mu_{n-1} \quad (50)$$

Now, using general node equation for nodes u_2 to $u_{\frac{n}{2}-1}$ and results from successor nodes we will obtain steady state equations in terms of β and μ_1 . Now, adding all such steady state equations of node u_1 to $u_{\frac{n}{2}-1}$, we have: $\mu_{\frac{n}{2}} - \mu_1 = -2\beta\left(1 + 2 + 3 + \dots + \left(\frac{n}{2} - 1\right)\right) + \left(\frac{n}{2} - 1\right)\mu_1$, so

$$\mu_{\frac{n}{2}} = -\beta \frac{n}{2} \left(\frac{n}{2} - 1\right) + \frac{n}{2} \mu_1 \quad (51)$$

Similarly, repeating the above procedure for nodes u_{n-1} to $u_{\frac{n}{2}+1}$, we have: $\mu_{\frac{n}{2}} - \mu_{n-1} = -2\beta\left(1 + 2 + 3 + \dots + \left(\frac{n}{2} - 1\right)\right) + \left(\frac{n}{2} - 1\right)\mu_{n-1}$, so

$$\mu_{\frac{n}{2}} = -\beta \frac{n}{2} \left(\frac{n}{2} - 1\right) + \frac{n}{2} \mu_{n-1} \quad (52)$$

Now, adding Eq.'s (51) and (52), we have: $2\mu_{\frac{n}{2}} = -\beta n \left(\frac{n}{2} - 1\right) + \frac{n}{2}(\mu_1 + \mu_{n-1})$. Using Eq. (50) and the fact that for Metropolis-Collect, we have $\mu_{\frac{n}{2}} \leq 1$ (see proof of Theorem 1) in above equation we get, $\beta \leq \frac{4}{n^2} = O\left(\frac{1}{n^2}\right)$. Hence, for cycle graph actual rate falls within our proposed bounds, i.e., neither of the bounds are tight. The situation is similar for the line.

5.5.2 Star graph with the sink at the center

Now, let G be a n node star graph with sink u_s at center. Now, for Metropolis-Collect algorithm with transition matrix \mathcal{P} , it is easy to check that second eigenvalue of given graph λ_2 is $\frac{n-2}{n-1}$. So, from Theorem 3 lower bound on stable data rate is $\beta \geq \frac{1-\lambda_2}{\sqrt{n(n-1)}} = \frac{1}{\sqrt{n(n-1)^3}}$.

Also, we know for the given graph transition probability $\mathcal{P}[u, u_s] = \mathcal{P}[u_s, u] = 1/\deg(u_s)$ for all $u \in V \setminus u_s$, so $\sum_{u \sim u_s} \mathcal{P}[u, u_s] = 1$. For given graph, we also have a vertex set $U = V \setminus u_s$ for which $h(U)$ is minimum, so $\hat{h}(G) = \sum_{u \sim u_s} \mathcal{P}[u, u_s]/n - 1$ for any $u \in U$. Using this result in Eq. (39) we get the upper bound for data arrival rate at nodes as, $\beta \leq \sum_{u \sim u_s} \frac{\mathcal{P}[u, u_s]}{n-1} = \frac{1}{n-1}$.

Now, to validate our rate bounds we again use the first principle method for finding stable rate of graph. Now, any node u_i other than sink only sends data and has no arrival as sink doesn't transmit any data, so steady state queue equation for node $u_i \in V \setminus u_s$ with $\mathcal{P}[u, u_s] = \mathcal{P}[u_s, u] = 1/\deg(u_s) = 1/n - 1$ is: $-\frac{\mu_i}{n-1} + \beta = 0$. As, for Metropolis-Collect, we have $\mu_i \leq 1$ (see proof of Theorem 1) so $\beta \leq \frac{1}{n-1}$. Hence, for star graph with sink at center upper bound on rate is tight.

5.5.3 Random geometric graph

Now let G be a random geometric graph where n nodes are thrown uniformly at random into a unit disc, we choose transmission radius r such that $r > (1 + \epsilon)r_c$ where $r_c = \Theta\sqrt{\frac{\log n}{n}}$ is the critical radius. This ensures that the graph remains connected with high probability [34, 16, 36]. The second eigenvalue corresponding to such graph for Metropolis-Collect (gossip setting) is $1 - \Theta(r^2)$ [36].

So, from Theorem 3 lower bound on the stable data arrival rate β is

$$\beta \geq \frac{1 - \lambda_2}{\sqrt{n(n-1)}} = O\left(\frac{\log n}{n^2}\right)$$

Now, we know $d_{max} = O(\log n)$ and $d_{min} = \Omega(\log n)$ for RGG (see Theorem 1.4 [22]). So, for any algorithm (not only Metropolis-Collect), with transition probability $\mathcal{P}[\cdot, \cdot]$, we have $\sum_{u \sim u_s} \mathcal{P}[u, u_s] \leq 1$ and from Proposition 1 and Eq. (39) upper bound for data arrival rate at nodes is given by,

$$\beta \leq \sum_{u \sim u_s} \frac{\mathcal{P}[u, u_s]}{n-1} \leq \frac{1}{n-1} = O\left(\frac{1}{n-1}\right)$$

Finding the actual rate from first principles like we did for other examples is difficult, so we skip it here, but we see in Section 6 through simulation that it appears the optimal rate is closer to the upper bound than the lower bound in this case.

6 Simulations

In this section we present our rate bounds for some commonly used graphs (Table 1) and discuss our simulation results.

6.1 Simulation set up

Our simulations have been written in C++ and done on a Core i7-4790, Quad-Core 3.6 GHz, 16GB RAM machine. To avoid any inaccuracies due to randomness, all results have been plotted after taking an average from three different simulation runs for each instance. We ran all the simulations for 100000 time slots to capture the packet transmission scenario and each simulation run took less than 2 minutes to execute. In our simulation results, we first analyze our algorithm's performance for Random Geometric Graph and d -regular Random Graph and then compare our algorithm with the Directed Diffusion, which is known for high data rate.

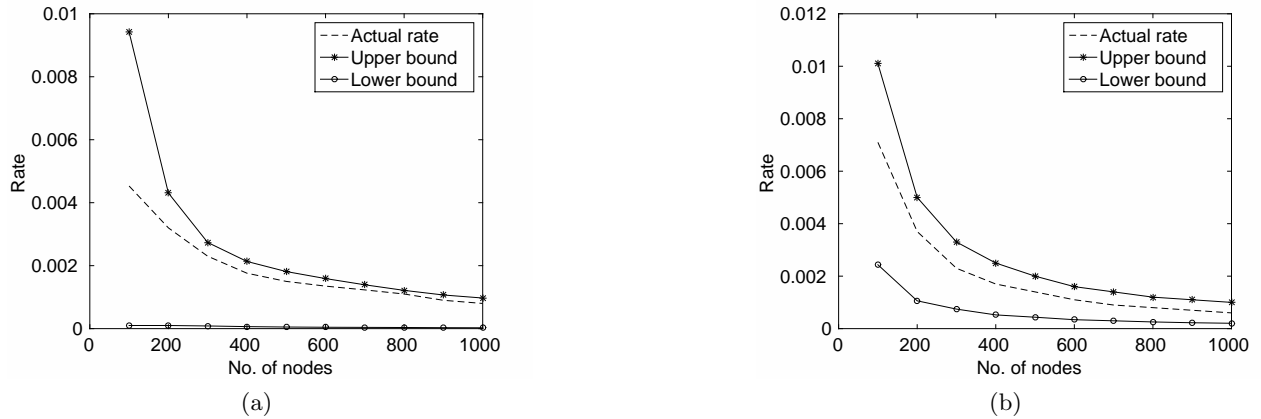


Figure 1: Performance analysis of Metropolis-Collect algorithm: Rate versus number of nodes (a) RGG (b) d -Regular Graph with $d=5$

Random Geometric Graph (RGG) is considered as the most appropriate model for sensor network [16, 27] as most of its applications require a random deployment of nodes. We perform our simulations on RGGs where nodes are thrown uniformly at random into a unit disc. We have taken care to use a radius that is large enough to ensure connectivity of the network [34, 16]. To further validate our results we perform simulations on d -regular random graphs.

For all our simulations we consider independent Bernoulli data arrivals with parameter β at all nodes. So, each node generates a data packet for each round and by the end of the simulation, different nodes may have generated different rounds of data. We equip each node with a queue which is used to store data packets which are not forwarded in given time slot. There is no priority given to any particular round's data packet in any of our simulations. We choose a node at the center as the sink. Our algorithm, Metropolis-Collect's performance is independent of sink location, centrally located sink is chosen to provide the best working environment for Directed Diffusion with which we compare our results. At every step, the sink absorbs any packet it receives. We say that a particular round of data is sunk when sink has received all the data packets of that round. As the data arrival rate β increases queues at nodes start filling up and *coupon collector problem* [28] steps in. Hence, the probability of data packets of earlier rounds being chosen for transmission is reduced, as a result of which the number of rounds sunk starts decreasing after particular rate resulting in infinite average collection time. So, our *stable data rate* β , as defined in section 4.1 is the rate which ensures finite average collection time $\bar{\tau}$ i.e. the maximum rate after which the number of rounds sunk starts decreasing. We also define *stability region* as the range of values of the data rate which ensures network stability. Metropolis-Collect's stability region lies within the bounds defined by Theorem 3 and Proposition 1.

6.2 Performance Analysis of Metropolis-Collect Algorithm

We compute the upper and lower bound for β using Theorem 3 for both RGG and d -regular graph with degree $d = 5$ for Metropolis-Collect algorithm. We also find the stable data rate β as defined in Section 6.1 and plot the rate versus number of nodes in graph. Figure 1a and Figure 1b validate our rate bounds as the stable rate lies within our bounds for both the graphs.

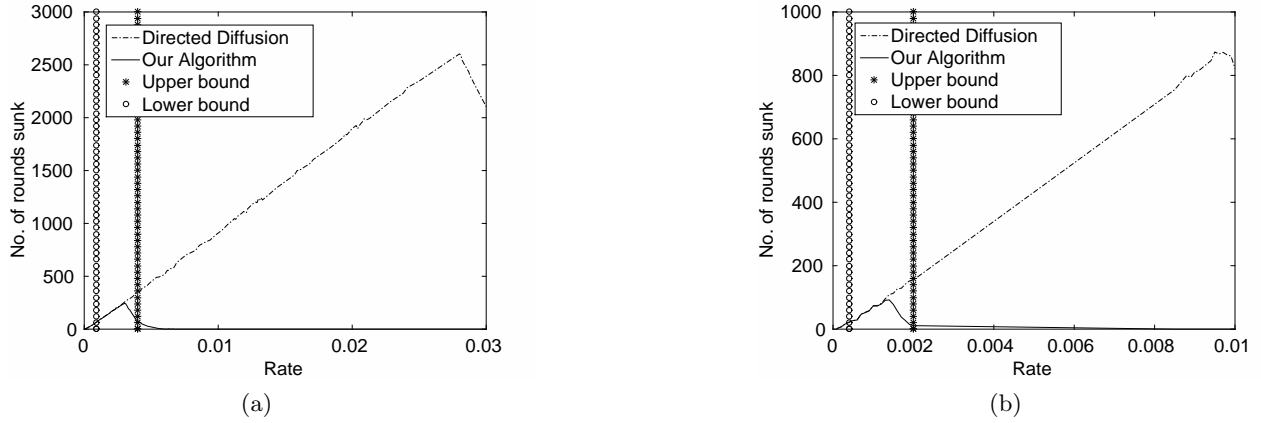


Figure 2: Performance comparison of Metropolis-Collect algorithm with Directed Diffusion: Number of rounds sunk versus rate (a) RGG with $n=200$ (b) d -Regular Graph with $d=5$ and $n=500$

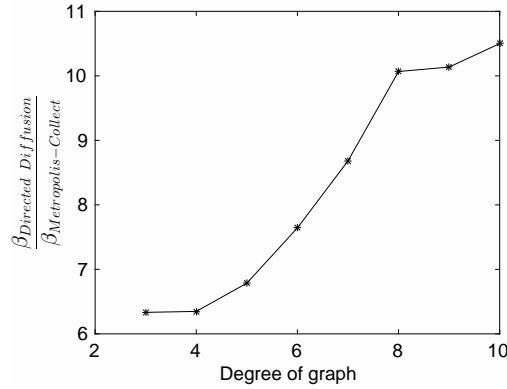


Figure 3: Variation of stable data rate ratio with degree of regular graph for $n=500$

6.3 Comparison with Directed Diffusion

In Directed Diffusion [21], based on the query and corresponding reply links between neighbors, sink reinforces an optimal low-delay path and each node randomly selects a packet from its queue and forwards it along the reinforced path. It is evident from Figure 2a and Figure 2b that though Metropolis-Collect's stability region is much smaller and corresponding throughput is lower but within Metropolis-Collect's stability region (indicated on graph between our lower and upper bound) its performance is comparable to that of directed diffusion. Also, in directed diffusion, every node deterministically forwards a data packet in each time slot, so $\beta \leq \frac{d_{us}}{n-1}$ whereas, in Metropolis-Collect $\beta \leq \sum_{u \sim us} \frac{\mathcal{P}[u, us]}{n-1}$. Since, in practical scenarios we have bounded degree networks advantage of directed diffusion over us is bounded. Figure 3 shows that the performance difference between two algorithms is directly proportional to degree of network. As the degree of nodes decreases, the ratio of the stable rate of Metropolis-Collect and that of directed diffusion also decreases.

7 Conclusion and Future Work

In this paper, we propose a simple decentralized and fault-tolerant Metropolis random walk based algorithm for data collection in a multi-hop sensor network. This algorithm is low configuration based and incurs no routing overhead. We analyze its latency and throughput under the assumption that each sensor node except the sink generates data as independent Bernoulli arrivals. Though our latency result is not comparable to routing based methods, it is just a logarithmic factor higher than the hitting time of a random walk on the network which is comparable to the worst case latency. We find bounds on network throughput given a stable data arrival rate that ensures finite average collection time. In particular, we find rate is lower bounded by the spectral gap of transition matrix at the heart of our algorithm. This lower bound is proportional to network's spectral gap for regular graphs. We also find a generalized upper bound for any data collection algorithm. We also generalize the rate result for m data sources and find a high probability bound on the average collection time of data arrival rounds and conclude that it is inversely proportional to the stable data arrival rate. Then, using simulations we compare our generalized algorithm with application-specific directed diffusion method and find that though our throughput is lower, the performance difference between two methods is proportional to the degree of network and also our performance is comparable to directed diffusion method in our stability region.

Incorporating interference and the full-duplex question As mentioned in Section 1, our model of the sensor network assumes simultaneous transmission-reception and multi-packet reception for each node in a given time slot. While these assumptions may be valid for full-duplex wired networks, they are not reasonable for wireless networks. However, it is important to note that if we were to move into an ALOHA-like situation where we allow for transmissions to fail due to the collision, both these assumptions are no longer required. Modeling this in a probabilistic setting is not difficult, and it is easy to see it will lead to an increase in latency which will grow with the degree of the network. There will be a small drop in throughput as well since the probability of reducing the queue (buffer) size (through successful transmission) will decrease. Both these changes can be incorporated in our model without changing the essential ideas underlying the proofs of our main theorems and so we omit them here.

As part of future work, analyzing Metropolis-Collect by considering branching random walks (currently we have a branching factor of 1) for data collection is an interesting line of work. We can also perform a similar analysis and mathematical abstractions for more sophisticated methods that use network coding like Growth Codes [23] to provide robust data collection in failure-prone sensor networks. We believe that our results form a baseline analytical framework that can be used to analyze average collection time in various settings.

Acknowledgment. The authors would like to thank Naveen Garg for some very useful discussions.

References

- [1] A. A. Abbasi and M. Younis. A survey on clustering algorithms for wireless sensor networks. *Comput. Commun.*, 30(14):2826–2841, 2007.
- [2] K. Akkaya and M. Younis. A survey on routing protocols for wireless sensor networks. *Ad Hoc Netw.*, 3(3):325–349, 2005.

- [3] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Comput. Netw.*, 38(4):393–422, 2002.
- [4] N. Alon, C. Avin, M. Koucký, G. Kozma, Z. Lotker, and M. R. Tuttle. Many random walks are faster than one. *Comb. Probab. Comput.*, 20(04):481–502, 2011.
- [5] C. Avin and C. Brito. Efficient and robust query processing in dynamic environments using random walk techniques. In *Proc. of the 3rd Intl. Symposium on Information Processing in Sensor Networks*, IPSN '04, pages 277–286, New York, NY, USA, 2004. ACM.
- [6] S. Banerjee, P. Gupta, and S. Shakkottai. Towards a queueing-based framework for in-network function computation. *Queueing Syst.*, 72(3):219–250, 2012.
- [7] R. J. Barton and R. Zheng. Order-optimal data aggregation in wireless sensor networks using cooperative time-reversal communication. In *Proc. of the 40th Annual Conf. on Information Sciences and Systems*, CISS '06, pages 1050–1055, March 2006.
- [8] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Gossip algorithms: design, analysis and applications. In *Proc. IEEE 24th Annual Joint Conf. of the IEEE Computer and Communications Societies*, volume 3 of *INFOCOM '05*, pages 1653–1664 vol. 3. IEEE, March 2005.
- [9] D. Braginsky and D. Estrin. Rumor routing algorithm for sensor networks. In *Proc. of the 1st ACM Intl. Workshop on Wireless Sensor Networks and Applications*, WSNA '02, pages 22–31, New York, NY, USA, 2002. ACM.
- [10] C.-K. Chau and P. Basu. Exact analysis of latency of stateless opportunistic forwarding. In *Proc. IEEE 28th Annual Joint Conf. of the IEEE Computer and Communications Societies*, INFOCOM '09, pages 828–836. IEEE, 2009.
- [11] F. R. K. Chung. Spectral Graph Theory. *Conference Board of the Mathematical Sciences*, (92):214, 1994.
- [12] R. Cristescu, B. Beferull-Lozano, and M. Vetterli. On network correlated data gathering. In *Proc. IEEE 23th Annual Joint Conf. of the IEEE Computer and Communications Societies*, volume 4 of *INFOCOM '04*, pages 2571–2582 vol.4. IEEE, March 2004.
- [13] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintenance. In *Proc. of the sixth annual ACM Symposium on Principles of distributed computing*, PODC '87, pages 1–12. ACM, 1987.
- [14] E. J. Duarte-Melo and M. Liu. Data-gathering wireless sensor networks: organization and capacity. *Comput. Netw.*, 43(4):519 – 537, 2003. Wireless Sensor Networks.
- [15] A. Giridhar and P. R. Kumar. Computing and communicating functions over sensor networks. *IEEE J. Sel. Area.*, 23(4):755–764, April 2005.
- [16] P. Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Trans. Inf. Theory*, 46(2):388–404, Mar 2000.
- [17] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *System sciences, 2000.Proc. of the 33rd annual Hawaii Intl. Conf. on*, HICSS '00, pages 10–pp. IEEE, 2000.

- [18] W. R. Heinzelman, J. Kulik, and H. Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *Proc. of the 5th Annual ACM/IEEE Intl. Conf. on Mobile Computing and Networking*, MobiCom '99, pages 174–185, New York, NY, USA, 1999. ACM.
- [19] S. Ikeda, I. Kubo, and M. Yamashita. The hitting and cover times of random walks on finite graphs using local degree information. *Theor. Comput. Sci.*, 410(1):94–100, 2009.
- [20] O. D. Incel and B. Krishnamachari. Enhancing the data collection rate of tree-based aggregation in wireless sensor networks. In *2008 5th Annual IEEE Communications Society Conf. on Sensor, Mesh and Ad Hoc Communications and Networks*, SECON '08, pages 569–577. IEEE, 2008.
- [21] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Trans. Netw.*, 11(1):2–16, Feb. 2003.
- [22] S. K. Iyer and D. Thacker. Nonuniform random geometric graphs with location-dependent radii. *The Annals of Applied Prob.*, pages 2048–2066, 2012.
- [23] A. Kamra, V. Misra, J. Feldman, and D. Rubenstein. Growth codes: Maximizing sensor network data persistence. In *Proc. of the 2006 Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '06, pages 255–266, New York, NY, USA, 2006. ACM.
- [24] B. Karp and H. T. Kung. Gpsr: Greedy perimeter stateless routing for wireless networks. In *Proc. of the 6th Annual Intl. Conf. on Mobile Computing and Networking*, MobiCom '00, pages 243–254, New York, NY, USA, 2000. ACM.
- [25] R. Karp, C. Schindelhauer, S. Shenker, and B. Vocking. Randomized rumor spreading. In *Foundations of Computer Science, 2000.Proc.. 41st Annual Symposium on*, FOCS '00, pages 565–574. IEEE, 2000.
- [26] S. Kashyap, S. Deb, K. V. M. Naidu, R. Rastogi, and A. Srinivasan. Efficient gossip-based aggregate computation. In *Proc. of the Twenty-fifth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '06, pages 308–317, New York, NY, USA, 2006. ACM.
- [27] H. Kenniche and V. Ravelomananana. Random geometric graphs as model of wireless sensor networks. In *Computer and Automation Engineering (ICCAE), 2010 The 2nd Intl. Conf. on*, volume 4, pages 103–107. IEEE, 2010.
- [28] D. A. Levin, Y. Peres, and E. L. Wilmer. *Markov chains and mixing times*. American Mathematical Soc., 2009.
- [29] S. Lindsey and C. S. Raghavendra. Pegasus: Power-efficient gathering in sensor information systems. In *Aerospace Conf. Proc., 2002. IEEE*, volume 3 of *AeroConf '02*, pages 3–1125. IEEE, 2002.
- [30] I. Mabrouki, X. Lagrange, and G. Froc. Random walk based routing protocol for wireless sensor networks. In *Proc. of the 2nd Intl. Conf. on Performance evaluation methodologies and tools*, VALUETOOLS '07, page 71. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2007.

- [31] T. Moscibroda. The worst-case capacity of wireless sensor networks. In *2007 6th Intl. Symposium on Information Processing in Sensor Networks*, IPSN '07, pages 1–10, April 2007.
- [32] D. Mosk-Aoyama and D. Shah. Computing separable functions via gossip. In *Proc. of the twenty-fifth annual ACM symposium on Principles of distributed computing*, PODC '06, pages 113–122. ACM, 2006.
- [33] M. J. Neely. Delay analysis for maximal scheduling with flow control in wireless networks with bursty traffic. *IEEE/ACM Trans. Netw.*, 17(4):1146–1159, Aug. 2009.
- [34] M. Penrose. *Random geometric graphs*. Number 5. Oxford University Press, 2003.
- [35] D. Puccinelli and M. Haenggi. Wireless sensor networks: applications and challenges of ubiquitous sensing. *IEEE Circuits Syst. Mag.*, 5(3):19–31, 2005.
- [36] D. Shah. Network gossip algorithms. *ICASSP, IEEE Intl. Conf. on Acoustics, Speech and Signal Processing - Proceedings*, pages 3673–3676, 2009.
- [37] S. Sternberg. *Dynamical systems*. Courier Corporation, 2010.

A Value of Appearance time factor Constant

Lemma 3. *For a constant $p > 1/2$, value of c satisfying the inequality $ec(1 - \beta)^c \leq p$ depends only on factor β .*

Proof. Let $p > 1/2$ be a constant, we will find a value of c which satisfies the inequality

$$ec(1 - \beta)^c \leq p$$

Above inequality can be rewritten as,

$$1 + \ln c + c \ln(1 - \beta) \leq \ln p \quad (53)$$

Since, $\ln(1 - \beta) = -\beta + \frac{\beta^2}{2} - \frac{\beta^3}{3} \dots$ for $-1 \leq \beta \leq 1$, so, $\ln(1 - \beta) \leq -\beta$ and we can write Eq.(53) as,

$$1 + \ln c - c\beta \leq \ln p \quad (54)$$

We know, if $a + b \leq c$ and $b \leq d$ then, $a + d \leq c$ implies $a + b \leq c$. So, value of c which satisfies Eq.(54) will also satisfy Eq.(53). Now, solving Eq.(54),

$$\begin{aligned} c\beta &\geq \ln c + 1 + \ln \frac{1}{p} \\ c\beta - \ln c &\geq 1 + \ln \frac{1}{p} \end{aligned}$$

For $c \geq e^\beta$, $c\beta - \ln c \leq (c - 1)\beta$. So, we need a solution to $(c - 1)\beta \geq 1 + \ln \frac{1}{p}$ such that the condition $c \geq e^\beta$ is satisfied. We have,

$$c \geq 1 + \frac{1}{\beta} \left(1 + \ln \frac{1}{p} \right) \quad (55)$$

$$c \geq 1 + \frac{b}{\beta} \quad (56)$$

where $b > 1$ is a constant. Since $\beta \leq 1$, Eq.(56) clearly satisfies the required condition and is only β dependent. \square